# **DAY 3 TOPICS**

Test plan

Creating a test plan is a crucial step in the software development process to ensure the quality and reliability of a product.

Here are six key steps to create an efficient test plan:

**Define Objectives and Scope:**

* Clearly articulate the objectives of the testing effort. What are the goals you aim to achieve through testing?
* Define the scope of testing by identifying the features, functionalities, or components to be tested.
* Ensure alignment with project requirements and stakeholders' expectations.

For example, Ensure that the e-commerce website functions correctly across different devices and browsers.

Whether it’s talking with the developers to understand the scope of the project, or working with a product manager to walkthrough new functionalities and user flow, defining the scope ensures that accurate information is being shared and that there is a common understanding of the product’s goals, expectations, and features.

**Schedule Timelines:**

Establish release deadlines, considering project timelines, past release schedules, development timeframes, and any external factors.

Example: Note every schedules and make sure it doesn’t coincide with release date.

* Release Deadline: March 15, 2024
* Project Timeline: January 1, 2024 - March 15, 2024
* Development Timeframe: January 1, 2024 - February 15, 2024
* Testing Timeframe: February 16, 2024 - March 10, 2024

#### **Define Test Objectives:**

Clearly state the reasons or purposes for testing, such as identifying defects, testing new features, or achieving specific levels of test coverage.

Examples of objectives for specific types of testing include:

**Functional Testing:**

* Objective: Ensure that the software functions as expected and meets the specified requirements.

Examples:

* + Validate user workflows, such as registration, login, and profile management.
  + Verify data processing functionality, including data input, manipulation, and output.

**Performance Testing:**

* Objective: Evaluate the system's responsiveness, scalability, and stability under various workload conditions.

Examples:

* + Measure response time for critical transactions and user interactions.

**Security Testing:**

* Objective: Identify vulnerabilities and weaknesses in the software to protect against potential security threats.
* Examples:
  + - Validate authentication and authorization mechanisms to prevent unauthorized access.

**Usability Testing:**

* Objective: Evaluate the software's user interface (UI) design and overall user experience (UX) to enhance usability and user satisfaction.
* Examples:
  + Test navigation and user flow to ensure intuitive interaction and ease of use.

#### Measure testing with the right metrics

[Metrics assess the overall quality of a release](https://www.testrail.com/), the progress of your testing, and the effectiveness of your testing (for a particular test cycle or the entirety of your testing).

They provide visibility into your testing process and overall product quality, ultimately helping your team decide if your release is ready to ship. Here are some metric formulas you might consider:

1. [Defect Density](https://www.geeksforgeeks.org/software-testing-defect-density/)
2. [Test Coverage](https://www.testrail.com/blog/test-coverage-traceability/)
3. Defect Detection Efficiency (DDE)
4. Time to Market

### **Determine test deliverables**

Identify the products of testing that track progress, meet project and client needs, and include documents like test plans, test suites, defect reports, and release notes.

* There are different test deliverables at every phase of the software development lifecycle.

#### **Before testing**

* **Test plan document:** The scope, objectives, and approach of the testing endeavor are all outlined in the test plan.
* **Test suite:**Test cases illustrate how to run a test, including input data, expected output, and pass/fail criteria.
* **Test design and environment specifications:** The test environment outlines the hardware and software configurations used for testing.

#### **During testing**

* **Test log:** The test log records each test case’s results, including issues and resolutions.
* **Defect report:** A defect report lists testing issues by severity, priority, and reproducibility.
* **Test data:** According to the International Software Testing Qualifications Board ([ISTQB](https://glossary.istqb.org/en_US/term/test-data-1-3)), test data is data created or selected to satisfy the execution preconditions and input content required to execute one or more test cases.
* **Test summary report:**The test summary report lists the number of tests run, passed, and failed, as well as open defects.

#### **After testing**

* **Test completion report:** Covers the testing scope, product quality, and lessons discovered.
* **User acceptance test (UAT) report:**Points to any issues found and fixed.
* **Release notes:** List information about what the release includes. Examples include any new features for development, advancements, or fixes.

**Design the Test Strategy:**

Determine the testing types, manual versus automated testing, risk documentation, test logistics, test criteria (suspension and exit criteria), and other considerations to guide the testing process.

Test strategy helps determine test cost, test effort, and which features will be in-scope.

1. Identifying testig types:
   * **Understand the Requirements**
   * **Consider Project Constraints**
   * **Analyze Test Objectives**
   * **Evaluate Testing Types**
   * **Identify Risks**
   * The complexity of your product
   * your team’s experience levels
2. Document risks and issues
3. Document test logistics
4. Establish test criteria:
   * + Test Criteria is a standard that regulates all activities within a testing project. The two main types of test criteria include suspension and exit criteria.

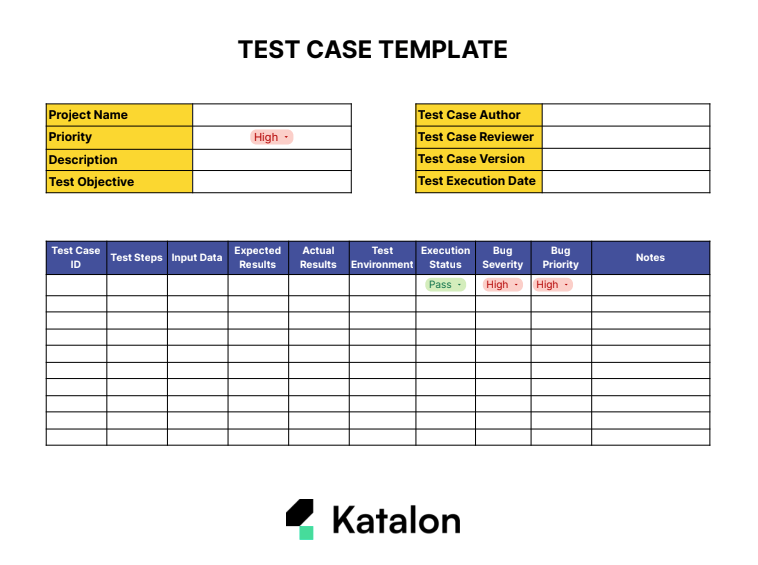
### **Plan the test environment and test data**

Set up the test environment, including hardware, software, network configurations, and test data preparation, ensuring it meets the requirements and is verified before testing begins.

1. Determine Hardware and Software Requirements:
2. Install Required Software
3. Configure the Network
4. Create Test Data
5. Access Builds:
   * Set up mechanisms to provide testers with access to the application builds they need to test.
   * Utilize file-sharing systems, version control systems, or other collaboration tools to ensure testers can access the latest builds conveniently.
   * Maintain proper version control to track changes and updates to the application builds.
6. Verify Test Environment:
   * Validate the test environment setup to ensure it meets the specified requirements and configurations.
   * Conduct preliminary tests to verify that all hardware, software, and network components are functioning correctly.
   * Address any discrepancies or issues identified during the verification process before proceeding with testing.

# **Test Case Template:**

* A test case is a specific scenario designed to verify the functionality and reliability of a software system. In the test case, testers outline the specific steps to be taken, the input data to be used, and the expected outcomes to determine if the software behaves as intended.
* Test cases are typically documented in a dedicated testing document (such as Google Sheets, or text document) or a test case management tool .For [automated testing](https://katalon.com/resources-center/blog/what-is-automation-testing), test cases can be written directly within code files using programming languages and testing frameworks.
* A test case template is a predefined structure that provides a standardized way to document and create test cases. It provides them as guidelines for testers to ensure that all necessary information about a test scenario is captured consistently across different test cases.
* With a good test case template, testing teams can maintain uniformity and improve the clarity and completeness of their test documentation.



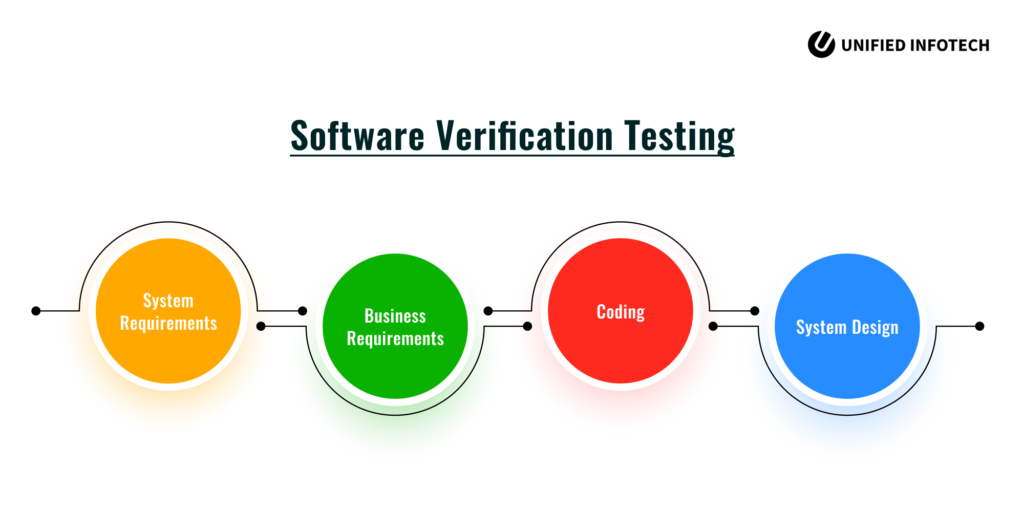
## **Best Practices To Write A Good Test Case**

1. Test Steps should be in detail and clear
2. **Use a Standardized Template**
3. **Understand the Requirements**
4. **Define Test Steps**
5. **Define Expected Results**
6. **Review and Validate**
7. **Provide Test Data**
8. **Prioritize Test Cases**

# Verification and validation in Testing

Verification

A test of a system to prove that it meets all its specified requirements at a particular stage of its development.



Verification is all about testing whether your software product satisfies the conditions you specified in the initial stages of development. It includes all actions taken to build the desired software product, like going through the specification documents, ensuring it complies with the code logic, checking code reviews, doing narrow inspections, design analysis, walk-through, static testing, etc.

The verification process eliminates bugs at the beginning stages of development to ensure the reliability of the product.

It also analyses how accurate and qualified the product design is.

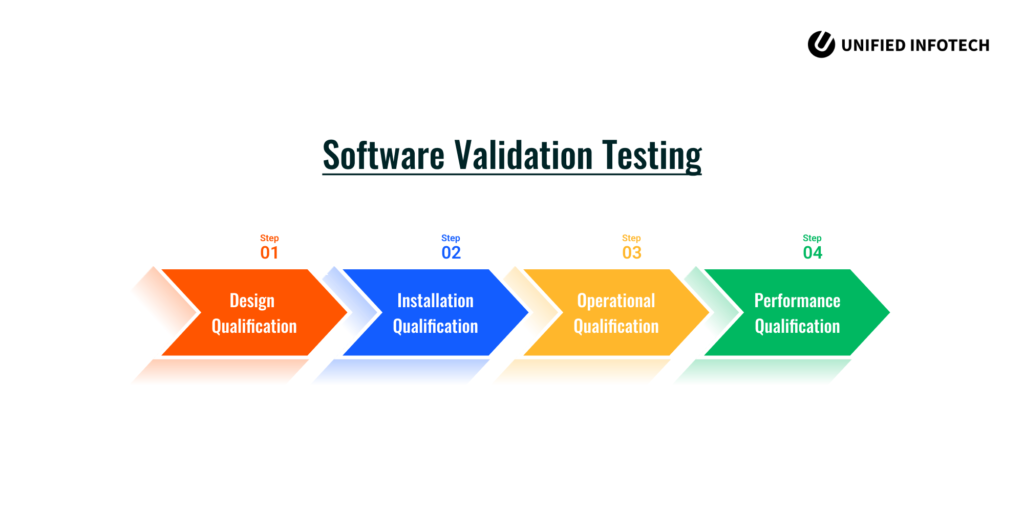
When to use???

Verification tests must be run at every stage of development before any feature is implemented.

* For example, consider a button labelled “Add to Cart”. Before creating this button, verification tests would review all relevant requirements previously decided in the ideation and brainstorming phases.
* Let’s say the documentation says the button must be black with the lettering in white. It should be no larger than 10mm X 10mm, and it should constantly be visible in the top right corner of every website product page. Another button with the exact text, color, and dimensions should be placed under every product on the page.
* Before creating the button, design and requirements documents must be reviewed, and all necessary specifications must be listed before work begins.
* Before working on every feature or element on the page, this must be done so the devs do not miss any guidelines.

Validation

An activity that ensures that an end product stakeholder’s true needs and expectations are met



Validation process is performed once a part of the product or the entire application is built completely.

Validation ensures that the necessary steps are taken to build the product as planned. Validation is generally carried out using [regression testing](https://www.pcloudy.com/a-brief-overview-of-regression-testing/), user testing, performance testing, etc.

Validation-related issues occur when there is improper or lack of communication between the user and the problem solver (developer). This is the reason why developers must use validation to check if the product that is built covers the user needs, their acceptance of the product, and ensures the proper functioning of the whole system to fulfill these needs. Usually, various companies go through acceptance testing by demonstrating their software product to the end-user and get their feedback to check how well they accept the product.

Most forms of QA fall under this category. All tests, from unit tests to User Acceptance Tests, are Validation tests.

**When to use???**

Validation tests must be run after every feature or step in the development process is completed.

* For example, [unit tests](https://www.browserstack.com/guide/unit-testing-a-detailed-guide), a form of validation tests, are run after every unit of code has been created.
* [Integration tests](https://www.browserstack.com/guide/integration-testing) are run after multiple modules have been completed individually and are ready to be combined.
* An essential element of validation testing is the running of [cross browser testing](https://www.browserstack.com/live). QAs must check that every function, feature, and design element appears and functions as expected on different browser-device-os combinations. For example, does the “Add to Cart” button work perfectly on Google Chrome running Samsung Galaxy A23 and Safari running on iPhone 13?

# V&V

A diagram of software development

Description automatically generated

 They both try to achieve two main tasks:

1. Check the product from the developer’s point of view: It checks if the product complies with the requirement. It also shows what a developer thinks about the final product.
2. Check the product from the consumer’s point of view: It checks if the product befits the customer’s use. It checks how a consumer looks at the final product.

* The difference between verification and validation is mainly in terms of the role that the specifications play.